# تقنية OpenCl

## إعداد : م.وسيم أبوزينة .


## خاص بمجلة الرقميات

## www. Alrakameiat.com

waz@alrakameiat.com


**التفرعية والحوسبة**

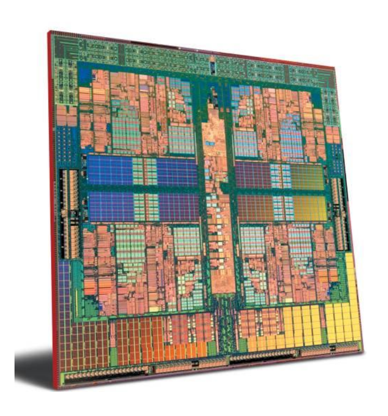
الحوسبة التفرعية هي شكل من اشكال الحوسبة التقليدية ولكن بفارق انه يتم القيام بعدة مُعالجات خلال واحدة الزمن " دورة معالج ، ثانية ، ...الخ" . يعتمد مفهوم الحوسبة التفرعية على ان مشكلة معينة يمكن ان يتم تقسيمها الى مشاكل جزئية اصغر ويتم معالجة المشاكل الجزئية على التفرع "التوازي " وتتميز المشاكل الجزئية بالاستقلال فيما بينها . يوجد العديد من انماط الحوسبة التفرعية : التفرع على مستوى البت bit ، على مستوى التعليمات ،على مستوى المعطيات وعلى مستوى المهام .ان مفهوم التوازي في عملية المعاجلة تم توظيفه لسنين عديدة في مجال الحواسيب عالية الاداء High Performance Computers او اختصاراً HPC ولكن الاهتمام بالتفرع اخذ يتزايد في الحواسيب العادية الاداء وذلك بسبب كسر حواجز ترددات عمل المعالج وغيرها من العوامل . فلقد امسى نموذج الحوسبة التفرعية النموذج المسيطر في معماريات الحواسيب بشكل معالجات متعددة النوى وذلك بالنسبة للحواسيب العادية وامسى نمط البرمجة التفرعي Concurrent Programming Methodology حاجة اساسية للتعلم.

يمكن تصنيف الحوسبة التفرعية استنادا الى المستوى الذي نتعامل معه على التفرع كما اننا نجد ان للحوسبة التفرعية اشكال عديدة فيمكن ان نجد تفرع مدعوم بشكل عتادي كالمعالجات متعددة النوى او عدة معالجات ضمن جهاز واحد او على مستوى المعالج الواحد فنحن نتكلم عن مفهوم ال Threads الذي يعطي الإيهام بان المعالج الواحد يعالج عدة امور بنفس الوقت كما نجد مفهوم الحوسبة على مستوى الاجهزة فنجد عدة حواسب تشكل ما يسمى بال Grid او الشبكة وما يتضمنه ذلك من تبادل للرسائل بين المعالجات .او التفرع المدعوم برمجيا وما يقابله من ادارة النياسب.

ان برامج الحاسب التي تعتمد على مفهوم الحوسبة التفرعية تتميز بصعوبة الكتابة اكثر من مثيلتها المكتوبة بكود تسلسلي ، لان التوازي بمعالجة المهام يقدم اصناف جديدة من الاخطاء البرمجية bugs .كما انه توجد مشاكل عديدة كمشكلة الاتصال بين المعالجات والمزامنة بينها كل هذا يعتبر من الصعوبات التي تواجه المبرمج عندما يريد كتابة برنامج بشكل تفرعي .كما يجدر الذكر بان السرعة المثلى لقياس زمن تنفيذ برنامج في حالة حوسبة تفرعية يمكن معرفته من قانون امبدال التالي :

$$S(p) = \frac{p}{(p-1)f+1}$$

حيث p عدد المعالجات و f الجزء اللازم من المسالة المراد حلها والذي لا يحل الا تسلسلياً .



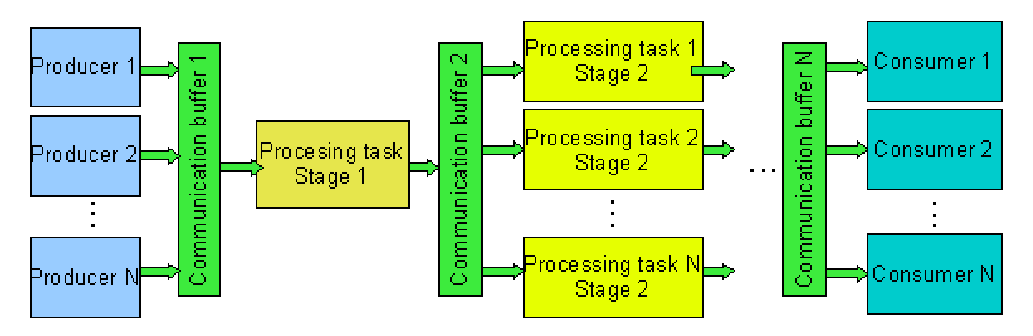
**التفرع او التوازي على مستوى المهام**

يعرف ايضا بالتفرع على مستوى التوابع والتحكم ، وهو شكل من اشكال التفرع للكود البرمجي بواسطة عدة معالجات في بيئة حوسبة تفرعية .ان التوازي على مستوى المهام يركز على المهام process او النياسيب المنفذة والموزعة على الحواسيب او المعالجات المختلفة .

في نظام يحوي عدة معالجات فان توازي المهام يمكن تحقيقه عندما يقوم كل معالج بتنفيذ مهمة (thread or process) بناء على معطيات واحدة او مختلفة .كما ان المهام المنفذة تتصل فيما بينها من اجل ضمان عدم تعارض المعطيات .

كمثال بسيط اذا كنا نشغل رماز مصدري على نظام يحوي معالجين ا وب في بيئة تفرعية ونحن نريد ان ننفذ المهام ا وب فمن الممكن ان نخبر المعالج ا بان يقوم بتخديم المهمة ا والمعالج ب بان يقوم بتخديم المهمة ب على التوازي ، وبالتالي ننقص من زمن تنفيذ البرنامج تقريبا الى النصف .

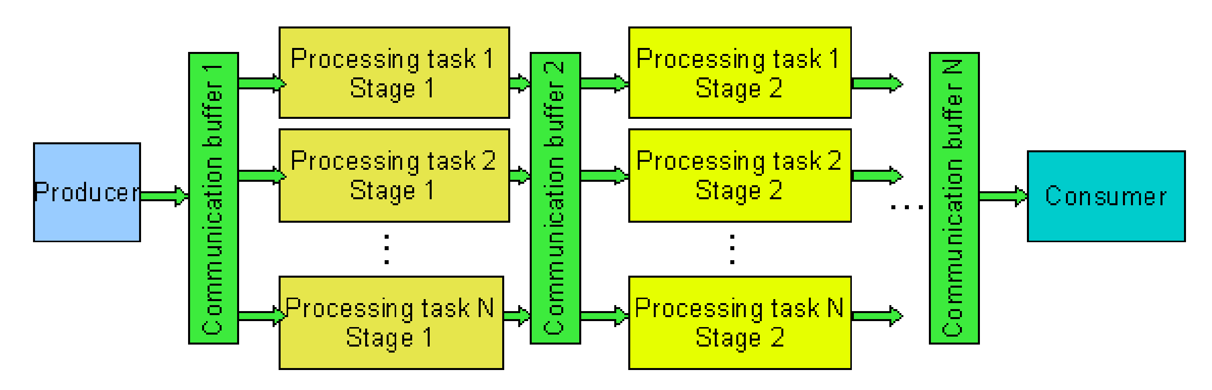
يركز التوازي على مستوى المهام طبيعة المعالج التفرعية على عكس التوازي على مستوى المعطيات .




**التوازي على مستوى المعطيات :**

ان التوازي على مستوى المعطيات يعرف ايضا بالتوازي على مستوى الحلقات هو شكل من اشكال التفرع في المعالجة في بيئة تحوي عدة معالجات .يركز التوازي على مستوى المعطيات على المعطيات الموزعة ما بين عقد المعالجة في البيئة التفرعية . في نظام يحوي عدة معالجات ينفذ مجموعة واحدة من التعليمات SIMD ،فان التوازي على مستوى المعطيات يمكن تحقيقه عندما يقوم كل معالج بالقيام بنفس المهمة على عدة كتل من المعطيات الوزعة .في بعض الحالات فان مهمة او نيسب واحد يتحكم بجميع مسار والعمليات المتعلقة بالمعطيات .

كمثال توضيحي : لنعتبر ان لدينا نظام يحوي معالجين المعالج ا والمعالج ب في بيئة تفرعية ونريد ان نقوم بمهمة محددة على بعض المعطيات د . من الممكن ان نخبر المعالج ا بان يقوم بمعالجة جزء من المعطيات د وان نخبر المعالج ب بان يقوم بمعالجة جزء من المعطيات الاخر من د على التوازي .وبالتالي ننقص من زمن التنفيذ . وكمثال آخر اعتبر انك تريد جمع مصفوفتين في تطبيق التوازي على مستوى المعطيات المعالج ا يمكن ان يجمع العناصر المتقابلة من المصفوفتين من الاعلى الى النصف بينما يقوم المعالج ب بالقيام بعملية الجمع من النصف الى اخر المصفوفة بين المصفوفتين بشكل متقابل . يؤكد التوازي على مستوى المعطيات على الطبيعة الموزعة للمعطيات كعملية معاكسة للتفرع على مستوى المهام .




**اغلب البرامج في العالم الحقيقي تكمن مابين التوازي على مستوى المهام ومابين التوازي على مستوى المعطيات .**

**مفهوم وحدة المعالجة البيانية للأغراض العامة GPGPU**

ان وحدة المعالجة البيانية للأغراض العامة General-purpose computing on graphics processing units (GPGPU او كما تعرف ب GPGP هي تقنية استعمال وحدة المعالجة البيانية GPU التي كانت حصرا تقوم بمعالجة الحسابات من اجل العمليات البيانية بالمعالجة العامة شانها شان وحدة المعالجة المركزية .كانت وحدة المعالجة المركزية CPU هي التي تقوم بمعالجة الحسابات الاخرى فيما مضى وحتى انها كانت تقوم بمعالجة العمليات البيانية من قبل ظهور GPU .

**ملاحظة :**

ان وحدات المعالجة البيانية GPU تعتبر بحد ذاتها وحوش معالجة فهي تحوي الكثير من انابيب المعالجة الصغيرة على عكس CPU الذي يحوي عددا اقل بكثير وبالتالي جاءت فكرة وحدة المعالجة البيانية للأغراض العامة لكي تستفيد من كلا من CPU& GPU بحيث نستفيد من ميزات كل منهما في عمليات المعالجة ولا تترك GPU فقط للأغراض البيانية .

**نشأة مفهوم GPGPU ؟**

**المرحلة الأولى - الحياة بسيطة**

قام الحاسوب على مفهوم Turing Machine ، وهو تصور جاءنا من الن تورينغ لجهاز الكمبيوتر .. خلاصته أن البيانات تنتقل من القرص الصلب إلى الذاكرة ، ومن يقوم بمعالجة تلك البيانات شيء اسمه معالج واختصاره هو .. CPU مرت ثلاثين سنة على هذه الفكرة والتي أصبحت هي المعتمدة في كل جهاز حاسوب تقريباً في مطلع ثمانينات القرن الماضي.

وفي العام 1990 ظهرت مكتبة ( أصبحت قياسية فيما بعد ) تحمل اسم OpenGL . المكتبة الرسومية استفادت لاحقاً من " كونها قياسية " ، وامسى كل صانع من صناع البطاقات الرسومية Graphics Card ، يعمل implementation خاص به ، يستفيد من قدرات العتاديات Hardware لتسريع عملية الرسوم . المبرمج في ذلك الوقت وجد أمامه مجموعة من " الدوال APIs " ، التي يستدعيها ليقوم بتغيير حالة متغيرات معينة ، فتقوم " بطاقة الرسوميات " بالاستجابة لهذه التغييرات .

البرامج الرسومية و الألعاب تصيب المعالج بإجهاد كبير ، وعملية نقل البيانات من المعالج إلى الذاكرة الرئيسية إلى البطاقة الرسومية ، تستهلك وقت وجهد ، بل أن النموذج السابق يحمل اسم Fixed-pipeline ، ويعني أن OpenGL كمكتبة رسومية قائمة على أمور ثابتة ومبرمجة مسبقاً ، بحيث تطلب من البطاقة الرسومية تغيير حالة ما .. فيظهر هذا على الشاشة .. فلا تستطيع مثلاً أن تتحكم بالمعادلة المسؤولة عن الشفافية .. فكل ما عليك أن تقوله هو : فعّل أو لا تفعّل. Enable Or Disable

**المرحلة الثانية - GPU**

ظهرت أفكار جديدة لعلاج مشاكل البرامج الرسومية و الألعاب ، وتتمحور حول مبدأ أساسي:
لماذا لا نضع معالج خاص + ذاكرة خاصة للبرامج الرسومية المبنية على OpenGL و أشباهها ؟
لماذا لا نسمح للمبرمج بكتابة معادلاته التي يريد تطبيقها على كل بكسل وعلى كل رأس Vertex ؟
فظهر بالتالي الوحدة لمعالجة البيانيات الرسومية Graphics Processing Unit او اختصاراً GPU وهو عبارة عن معالج مركزي خاص للمكتبات الرسومية مثل CG و OpenGL و Direct3D .يوجد ضمن كرت الشاشة او الاظهار بهدف تقليل عمليات المعالجة المكلفة على وحدة المعالجة المركزية CPU .وهذا لا يعني الاستغناء عن CPU فمثلا عملية ضرب المصفوفات وعملية معالجة كل بكسل ومعظم العمليات التي تختص في عمليات الغرافيكس امست تعالج في معالج كرت الاظهار لا معالج لوحة الام CPU .

**المرحلة الثالثة GPGPU**

ظهرت فكرة وحدة المعالجة المركزية للأهداف العامة من قوة معالجات كروت الاظهارGPU في المعالجة فبدلا من جعل GPU مخصص فقط لمعالجة عمليات الغرافيكس المتطلبة لم لا يتم الاستفادة منها في جوانب تطبيقية اخرى .
من هنا ظهر مفهوم : GPGPU وهو اختصاراً لـ .. General Purpose Graphics Processing Unit يعني " الغايات العامّة من وحدة المعالجة الرسومية " .. فظهرت مقالات وتطبيقات تستفيد فعلاً من إمكانية GPU ، لم ننتظر طويلاً حتى ظهرت CUDA و OpenCl و مكتبة Direct Compute .

**المرحلة الرابعة OpenCL وCUDA**

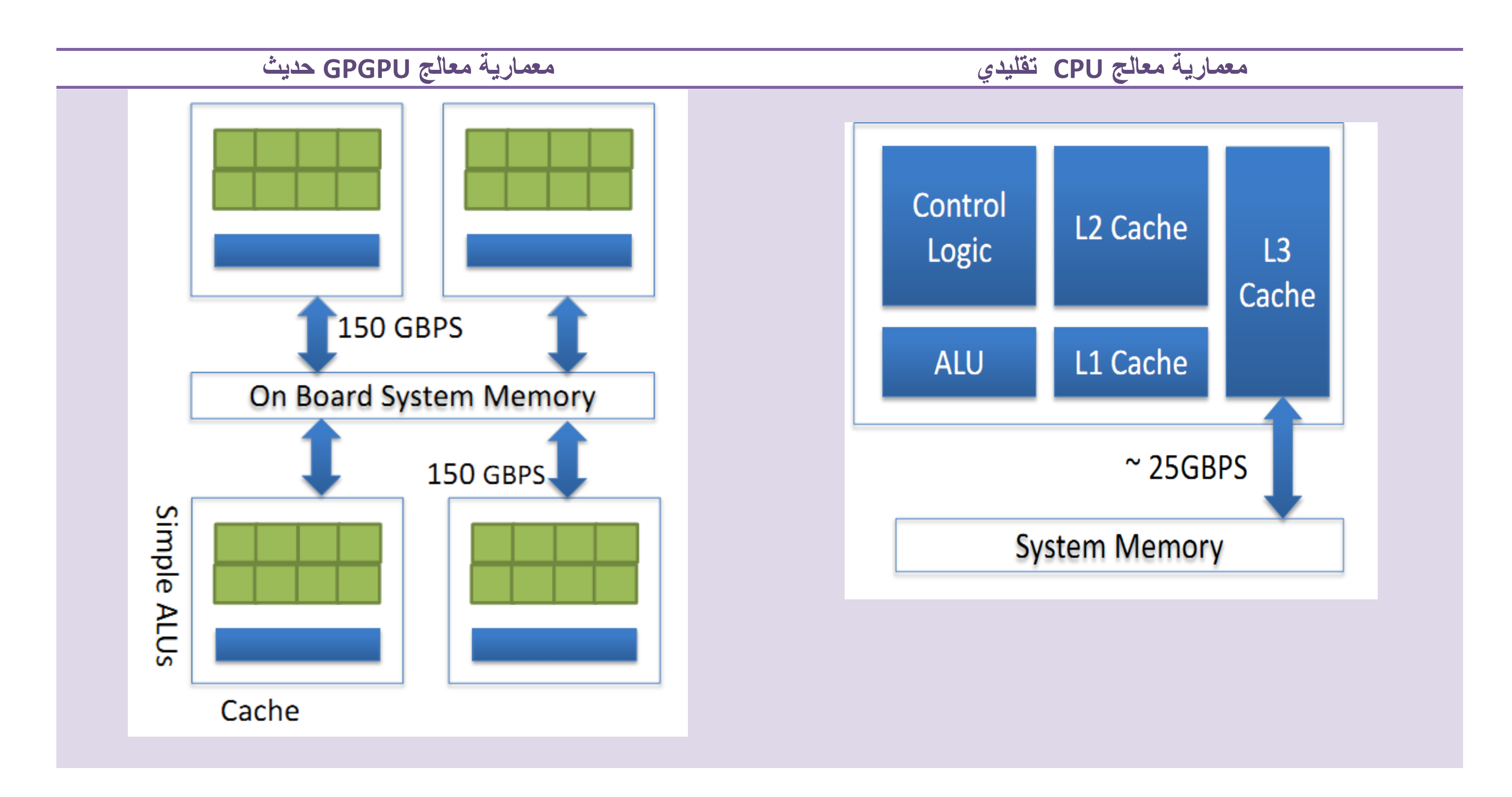
ظهرت الحاجة لمكتبة قياسية تستفيد من العدد الضخم من الأنوية cores وبالتالي تطبيق مبدأ البرمجة المتوازية ، بل والاستفادة من قدرات GPU الحسابية السريعة و الدقيقة ، حتى لو لم يكن التطبيق رسومي.

فقد قام مصنعو البطاقات الرسومية بتطوير قدرات GPU ، ليخدم قطاعات أخرى ، ولعل خطوة شركة انفيديا NVidia في هذا المجال هي الأبرز من حيث تطوير CUDA وهي معمارية ظهرت على يد شركة انفيديا NVidia وتعمل على بطاقات انفيديا الرسومية ، و يمكن من خلالها تطوير تطبيقات لا علاقة لها بالرسوميات ( ويمكن استخدامها طبعاً مع التطبيقات الرسومية ) ، وطبعاً يمكن الاستفادة من هذه المعمارية من خلال لغة C ولغات أخرى.

ومن ثم ظهرت مكتبة OpenCL من مختبرات Apple والتي أصبحت الآن تحت عهدة Khronos نفس الجهة التي تشرف على مواصفات OpenGL ، وبالتالي أصبحت مكتبة قياسية يشارك في وضع قياساتها ومن ثم عمل implementation لها ، عدة شركات ضخمة ,AMD, IBM, Intel و Nvidiaبالإضافة لـ Apple طبعاً.

تتميز OpenCl بقدرتها على العمل على أي قطعة, hardware التي تسمى Device ، فهي قد تعمل على CPU وتستفيد من .. multi-core CPU, وقد تعمل علىGPU . علماً أن عدد " أنوية " المعالج المركزي CPU قد تصل لثمانية أنوية أو أقل من هذا أما في ال GPU فأنت أمام عشرات ومئات الأنوية التي تتسابق لخدمتك .
OpenCl و CUDA لهما نفس الهدف تقريباً. ولكن التوجه الآن هو بدعم OpenCl بشكل كامل ، باعتبارها قياسية وستعمل على أكثر من بطاقة رسومية وعلى أكثر من نظام تشغيل ، بل وحتى على أكثر من لغة برمجية . بالتالي أصبحنا بغنى عن التطفل على مبرمجي الرسوميات واستخدام أدواتهم مثل لغة GLSL أو HLSL لاستغلال قوّة GPU في أمور غير رسومية ، بل هناك ما يسمى بـ kernel في OpenCl وهي لغة خاصة صممت لهذا الغرض ( مبنية على مواصفات لغة سي ).

# CPU Vs GPGPU




**معمارية معالج CPU تقليدي**

1- **يتم تخصيص مساحة لا بأس بها لقسم التحكم CL او control Logic عوضا عن تخصيص مساحة اكبر لقسم المنطق والعمليات الحسابية ALU.**
2- **يوجد عدة مستويات للخبيئة تستعمل للتقليل من التاخير الزمني الحاصل من عمليات النقل.**
3- **عدد محدود من السجلات بسبب عدد النياسب Threads المفعلة القليلة نسبياً.**
4- **يهتم بالتاخير الزمني لنيسب واحد .**
5- **التحكم بالمنطق لاعادة ترتيب تنفيذ التعليمات**
6- **يتم تحسينها من اجل عمل مهمة واحدة**
7- **تسلسل تنفيذ التعليمات غير مرتب حسب الورود، كما ان التنبؤ بالتفرع والقفز والخابية تاخذ معظم حجم الشريحة .**

**معمارية معالج GPGPU حديث**

1- مساحة اقل مخصصة لكتلة التحكم والمنطق والكاش
2- تحوي عدد كبير من الانوية البسيطة cores
3- يحوي العديد الانوية المعالجة.
4- ملف سجلات اكبر لدعم عدد اكبر من النياسب
5- زمن تاخير قليل بسبب دعم عتادي الذي ينظم التبديل بين النياسب
6- عدد كبير من وحدات الحساب ALU لكل نواة core مع عدد محدد من الخابيات "cache " لكل نواة .
7- ممر الذاكرة محسن من اجل تحسين عرض الحزمة

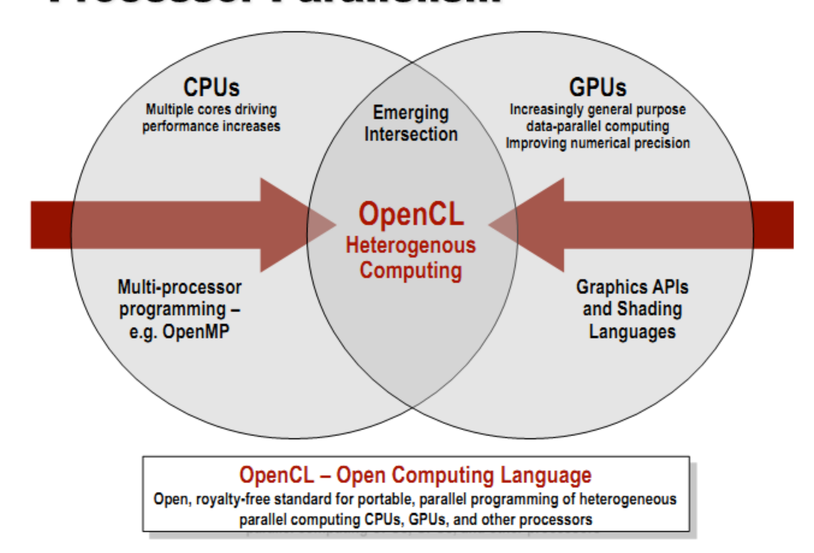
## ما هي تقنية  OpenCl

ان OpenCl "Open Computing Language" هي منصة عمل " كما يمكننا القول بانها معايير مواصفات specification "مخصصة لكتابة برامج التي تنفذ على منصات هجينة تحوي وحدة معالجة مركزية CPU كما تحوي وحدة معالجة بيانات GPU ومعالجات اخرى .يهدف هذا المعيار الى كتابة برامج تستفيد من القوة الحسابية للمعالجات سواء كانت معالجات CPU او GPU او حتى معالجات الآلة الحاسبة البسيط وتهدف بشكل آخر الى استثمار موارد الحاسب الى اقصى حد ممكن .تقدم OpenCl حوسبة تفرعية من خلال مفهومي التوازي على مستوى المهام وعلى مستوى المعطيات . وتم تضمين معايير OpenCl في كروت الاظهار من قبل كلا AMD/ATI باسم Stream SDK، وقامت إنفيديا NVidia بتقديم OpenCl كمثيله CUDA "Compute Unified Device Architecture " وقامت مايكروسوفت بتضمين هذه المعايير باسم DirectCompute.

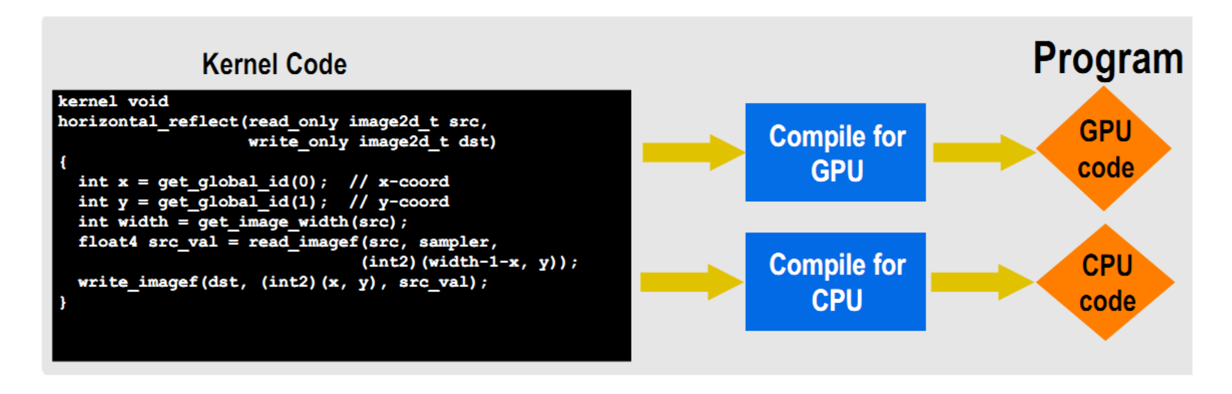
تضمن OpenCl لأي تطبيق بإمكانية استغلال وحدة المعالجة البيانية GPU لأغراض لا تتعلق بالإظهار والغرافيكس .وهكذا يقوم OpenCl باستغلال امكانيات كرت الاظهار الضخمة جدا في عمليات المعالجة للمعالجة العامة التي لا تختص الغرافيكس فحسب .

يوجد تشابه كبير بين OpenCl وتقنيات مثل OpenGL و OpenAL كمعايير صناعية .ويتم ادارة هذه المعايير من قبل مجموعة Khronos.كما يجدر الذكر بان شركة آبل هي من قامت باقتراح هذا المعيار .

الصورة التالية توضح ان معيار OpenCl هو معيار صناعي وبرمجي نتج عن التقاطع بين امكانية معالجة وحدة المعالجة المركزية وبين امكانية معالجة وحدة المعالجة البيانية .




الشكل التالي يوضح آلية عمل OpenCl بشكل مبسط :



## ماهي الفكرة الجديدة التي جاء بها معيار OpenCl ؟

تم استبدال الحلقات loops بالتوابع (نواة Kernel) بحيث يتم تنفيذ هذه التوابع كل في مجاله . مثال لنفرض اننا نريد معالجة صورة 1024*1024 بتشغيل نواة واحدة لكل بيكسل او 1048576 عملية تنفيذ للانوية بتقسيم معين لكل بكسل .عندها فان كل نواة ضمن ال GPU يمكن ان تفيد بهذه العملية وبالتالي فارق الزمن هائل بين التنفيذ التسلسلي والتنفيذ على التوازي .يجدر الذكر ان وحدة المعالجة البيانية تحوي عدد ضخم من الانوية المعالجة .

**Traditional loops**

```
void
trad_mul(int n,
         const float *a,
         const float *b,
         float *c)
{
  int i;
  for (i=0; i<n; i++)
    c[i] = a[i] * b[i];
}
```

**Data Parallel OpenCL**

```
kernel void
dp_mul(global const float *a,
       global const float *b,
       global float *c)
{
  int id = get_global_id(0);

  c[id] = a[id] * b[id];

} // execute over "n" work-items
```

## لماذا OpenCL ؟

1- القوة الحسابية انتقلت من سرعة الساعة الى النوى Cores .بعدما وصلت ترددات ساعة المعالجات الى طريق مسدود تقريبا لذا تم الاعتماد على انشاء عدة انوية تخدم المتطلبات الكبيرة للمعالجة والحساب .

2- بسبب توفر العديد من المعالجات CPU و الرغبة بوحدة معالجة بيانية GPU قابلة للبرمجة .

3- الحاجة الى واجهة برمجية التي تسمح للمستخدم باستغلال الموارد بشكل سهل .

4- دعم البرمجة التفرعية للأغراض العامة .

5- مصممة لتكون Device agnostic اي يمكن لأي تطبيق للمعايير ان يعمل على اي معالج دون آخر .

6-كمعيار مفتوح يجب ان يكون الرماز المصدري البرمجي يجب ان يتميز بالمحمولية بان يعمل على نظام ويندوز وابل وغيره .

7-لاتوجد شركة محددة تتحكم بوضع المعايير .

## الشركات الداعمة للمعيار :






































**الاجهزة التي تدعم OpenCl**

غالبا ما يتم التعامل مع وحدة المعالجة المركزية CPU ووحدة المعالجة البيانية GPU ولكن يمكن التعامل مع اي جهاز يدعم المواصفات الدنيا من المعايير فيمكن ان يكون Multimedia Chip,FPGA,Embedded processors واي جهاز يدعم هذه المعايير .فأقوى ميزة في هذا المعيار هو انه يمكنك ان تكتب برنامجك بالشكل الطبيعي ,وتجد انه يعمل على اي جهاز يدعم وحدة لمعالجة المعطيات .

**الهدف من OpenCl**

1-يهدف هذا المعيار ان يكون نموذج بسيط للحساب ومكتبة برمجيةAPI سهلة التعامل
2-دعم لغة ANSI-C99 بحيث تم اضافة انماط معطيات وتوابع اخرى .
3-بيئة عمل لإدارة النياسب Thread management framework فليس عليك ان تهتم بإنشاء او حذف النياسب فمنصة OpenCl ستقوم بهذا العمل نيابة عنك .
4-سهل الاستعمال ويجب ان يكون خفيف على الجهاز وفعال .
5-الحاجة الى استعمال جميع الموارد الحسابية في الكمبيوتر .
6-اقل قدر من الاخطاء للتوابع الرياضية بحيث نعلم ان المعالج بشكل عام يقوم ببعض الاخطاء فيما يتعلق بأمور كالفاصلة العائمة بسبب طريقة ترميز الاعداد .

**استعمالات OpenCl**

1-معالجة الصورة ، الفيديو والصوت .
2-عمليات المحاكاة والحسابات العلمية .
3-التصوير الطبي : والتي تملك كم ضخم جدا من البيانات التي تحتاج للمعالجة .
4-النماذج المالية والبورصة. بحيث توجد العديد من المعطيات المالية المتغيرة كل ثانية وتحتاج الى معالجة كبيرة لكي تكون up to date

**OpenCl & OpenGL**

OpenCl صمم ليكون متوافق بشكل كبير مع OpenGL بحيث يمكن تبادل المعطيات الناتجة عن الحسابات من OpenCl الى مكتبة الاظهار بحيث تتكامل المكتبتين ما بين الاظهار والمعالجة لتسريع عمل البرنامج .

**ما الحالات التي لا تناسب استعمال OpenCl**

1-شأنها شان اي مسالة تفرعية في حال كانت المسالة المحلولة لاتحل بطريقة تفرعية وانما بشكل تسلسلي بحت فنجد ان OpenCl لن يؤثر على الاداء بشكل فعال .

2-الحسابات التي تتطلب العديد من المؤشرات المترابطة التي تتعامل كثيرا مع حالات كالدخل والخرج (تعامل مع القرص الصلب مثلا ) .

3-الحسابات التي تتطلب الكثير من الاتصال والارتباط ونتائج التحديث بحيث خرج الاول هو دخل للثاني بحيث نريد ان نجد معطيات مستقلة بأكبر شكل ممكن كي نستطيع المعالجة بسرعة .

## بنية برنامج OpenCl

يقسم برنامج OpenCl الى قسمين

1- كود المضيف Host
   - رماز مصدري بلغة ++c/c الذي سيعمل على الcpu ، يتم ترجمته باستعمال المترجمات التقليدية + OpenCl Headers
   - يستعمل OpenCl API من اجل :
     - نقل المعطيات من ذاكرة النظام الى ذاكرة كرت الاظهار GPU DRAM
     - يبدأ بتشغيل عدة نسخ من النواة ليعمل على ال GPU
     - كل نسخة مولدة تعمل على جزء من المعطيات المطلوبة
     - تعيد المعطيات بعد نسخ النتائج
2- كود النواة Kernel
   - رماز مصدري بلغة c الذي سيعمل على GPU ، يتم ترجمته باستعمال مترجمات خاصة وفق الشركة المنتجة مثل مترجم إنفيديا.
   - يتعامل مع المعطيات المخزنة في ذاكرة كرت الاظهار
   - يكتب النتائج في ذاكرة كرت الاظهار GPU DRAN.

## نموذج إنفيديا أوبن سي إل NVidia OpenCL

ميزات نموذج إنفيديا NVidia 


1- دعم انظمة التشغيل :
   - 32 and 64 bit Windows XP and Vista (and soon Win 7)
   - 32 and 64 bit Linux (Ubuntu, RHEL, etc.)
   - Mac OS X Snow Leopard (indirectly via Apple)

2- بيئة التطوير IDE :
   - Visual Studio 2005 (8) and Visual Studio 2008 (9) for Windows
   - GCC for Linux
3- برامج التعريف والكومبايلر
   - In SDK for Alpha & Beta
   - يتم توفير برامج التعريف ضمن قرص تعريف كرت الاظهار
4- SDK
   - Source code & white papers for sample applications (30 presently)
   - Documentation: Getting Started Guide, Programming Manual,
   - Best Practices (Optimization) Guide